illumina®

HiSeq Analysis Software v2.1 User Guide

For Research Use Only. Not for use in diagnostic procedures.

ILLUMINA PROPRIETARY

Document # 15070536 v01
January 2017

Customize a short end-to-end workflow guide with the Custom Protocol Selector
support.illumina.com/custom-protocol-selector.html

HiSeq Analysis Software v2.1

Software Guide

Introduction

HiSeq Analysis Software v2.1 is a software package for analyzing sequencing data generated by Illumina HiSeq X sequencing systems. The software leverages a suite of proven algorithms to detect genomic variants comprehensively and accurately. HiSeq Analysis Software v2.1 is a complete package with a range of variants, including Single Nucleotide Variants (SNV), Indels, Structural Variants (SV) and Copy Number Variants (CNV) for tumor and normal samples. HiSeq Analysis Software v2.1 utilizes the base calls and quality scores generated by Real-Time Analysis (RTA) software during the sequencing run to rapidly handle data for high-throughput whole-genome sequencing analysis.

HiSeq Analysis Software v2.1 uses the Isaac Aligner and Strelka Germline Variant Caller to provide both aligned and unaligned reads and variants. See *Isaac Aligner* on page 21 and *Strelka Germline Variant Caller* on page 22 for more information.

For structural variants, HiSeq Analysis Software v2.1 uses 2 complementary approaches:

- Read depth analysis by Canvas. See *Canvas* on page 29.
- Discordant paired-end analysis by Manta. See *Manta (Structural Variant Caller)* on page 33.

HiSeq Analysis Software v2.1 supports a multistep analysis workflow that uses a simple command line and recommended default settings specified in the sample sheet.

This document provides an overview of the HiSeq Analysis Software v2.1 workflow, data structure, computing requirements, and installation guidelines. The aim of this document is to help you understand the details and use of HiSeq Analysis Software v2.1 for high-throughput whole-genome sequencing analysis.

Computing Requirements

Operating System

HiSeq Analysis Software v2.1 requires the CentOS 6 operating system with docker v1.62 or later. To run docker on CentOS-6.5 or later, kernel 2.6.32-431 or higher is required.

To install docker on CentOS 6, enter the following commands:

```
yum -y install epel-release
yum -y install docker-io
```

Compute Hardware

HiSeq Analysis Software v2.1 requires specific compute architecture for proper functionality and performance. The hardware required is available from a range of suppliers and manufacturers.

For a HiSeq X Ten system, the following architecture is recommended for analysis. Archival storage is not included in this recommendation.

- Resilient NAS solution serving CIFS and NFS
  - A minimum of 100 TB usable storage capacity with performance > 2 GB/s
    To retain data after analysis, additional capacity is needed.
  - Multiple 10 GB or 40 GB network links
- Resilient 10 GB network
- 26 servers, each with the following:
  - Dual 10 Core CPUs @ 2.8 GHz
  - 128 GB 1867 MHz memory
  - 6 TB usable local storage capacity with performance > 500 MB/s

- 10 GB Ethernet adapter

Install HiSeq Analysis Software Docker Image

To perform installation and extract the reference genome, you need root privileges.

1 Download the following from the link provided by Illumina.
   - isis-2.6.53.23.tar md5sum 3cb4dbc3853bd8a856b2b84c0dc6cec6
   - hg38-NSv4.2.tgz md5sum 435cdc86c8ec31135f0e7443f9344e70
   - hg19-NSv md5sum a08c2f68a3c6a90cb0d3cd0cf713c162

2 Install the analysis software:
```
cat isis-2.6.53.23.tar|docker load
```
Verify the installation:
```
sudo docker images
```

3 Extract the hg19 and hg38 reference genomes:
```
tar -zxvf hg19-NSv4.2.tgz -C /genome
tar -zxvf hg38-NSv4.2.tgz -C /genome
```

When running HiSeq Analysis Software on a cluster, create a script with the necessary command lines and schedule the script on a node. See your scheduler documentation for the correct command.

Make sure that the node is dedicated to the script. When running, HiSeq Analysis Software uses all the RAM and cores on the node.

To run sudo docker, a password is not needed.
```
sudo visudo
USERNAME ALL = NOPASSWD:/
```

To run docker without sudo, create a docker group and run the docker daemon as part of that group. Then add all HiSeq Analysis Software users to the group.

Run HiSeq Analysis Software in Docker with Open Grid Scheduler

Since the docker command to run HiSeq Analysis Software does not specify the number of cores or amount of memory that the analysis is limited to, the analysis uses the entire node and all RAM available. With a parallel environment, the analysis is sent to only 1 node. The analysis fully occupies that node to prevent another job from using the node and oversubscribing that node. For an example of a threaded parallel environment, see the Oracle entry *Configuring a New Parallel Environment*.

Use Open Grid Scheduler to create a parallel environment. The following command assumes that docker can be run without sudo.

NOTE:

The command specifies 20 cores per node. If you enter > 20 cores, the job does not run. If you enter < 20 cores, there is a risk that other jobs use the same node and oversubscribe that node.

```
qsub -pe threaded 20 -cwd -b y -V /path_to_runhas.sh/runhas.sh
```

Where `runhas.sh` is
```
#!/bin/sh
# each node may not have isis-2.6.53.23 images loaded.
cat /PATH_2_HAS_TAR/isis-2.6.53.23.tar|docker load
docker run -i \
-v /etc/localtime:/etc/localtime:ro \
```

```
-v /PATH_TO_RUN/150101_E00001_0001_AHNN7NCCXX/:/Run:rw \
-v /genome/:/genome \
illumina/isis:2.6.53.23 mono /opt/illumina/Isis/Isis.exe -r /Run -a
   /Run/HAS21
```

Threaded environment:

```
pe_name threaded
slots 9999
user_lists NONE
xuser_lists NONE
start_proc_args /bin/true
stop_proc_args /bin/true
allocation_rule $pe_slots
control_slaves FALSE
job_is_first_task TRUE
urgency_slots min
accounting_summary FALSE
```

GenerateFASTQ Workflow

The first items to process in the GenerateFASTQ workflow are the run folders with base call and quality score data in *.bcl.gz file format. The workflow demultiplexes the data and converts it to FASTQ files organized under a per-sample directory structure.

Initially, you create a SampleSheet.csv file for the GenerateFASTQ workflow step. A new sample sheet file is automatically generated for each step, specifying the settings that will be applied in subsequent steps. For more information, see *Sample Sheet* on page 20.

Run GenerateFASTQ

1 Identify a run folder for the analysis, which can be the original run folder for the flow cell.

2 In the top level of the run folder, create a sample sheet named SampleSheet.csv, as shown in the example below. Make sure that the GenomeFolder location is correct.
For more information, see *Sample Sheet* on page 20.

```
[Header],,,,,,,,,
IEMFileVersion,4,,,,,,,,
Investigator Name,S. w.,,,,,,,,
Experiment Name,,,,,,,,,
Date,2/12/2015,,,,,,,,
Workflow,GenerateFastq,,,,,,,,
Application,HiSeq FASTQ Only,,,,,,,,
Assay,TruSeq HT,,,,,,,,
Description,HiSeqX run,,,,,,,,
Chemistry,Default,,,,,,,,
,,,,,,,,,
[Reads],,,,,,,,,
151,,,,,,,,,
151,,,,,,,,,
```

```
,,,,,,,,,
[Settings],,,,,,,,,
ReverseComplement,0,,,,,,,,
Adapter,AGATCGGAAGAGCACACGTCTGAACTCCAGTCA,,,,,,,,
AdapterRead2,AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT,,,,,,,,
,,,,,,,,,
[Data],,,,,,,,,
Lane,SampleID,SampleName,Sample_Plate,Sample_Well,I7_Index_
   ID,index,GenomeFolder,Sample_Project,Description
8,NA12878_GiaB1_ID,NA12878_GiaB1_Name,,,,,/genome/Homo_
   sapiens/UCSC/hg19/Sequence/WholeGenomeFasta,Project,
```

3 Start the GenerateFASTQ workflow for each flow cell. Specify absolute paths. Relative paths are not supported. Example:

```
sudo docker run -i -t \
-v /etc/localtime:/etc/localtime:ro \
-v /Runs/150724_E00315_0078_BH5NWKCCXX /:/Run:rw \
-v /hg19-NSv4.2/:/genome \
illumina/isis:2.6.53.23 mono /opt/illumina/Isis/Isis.exe -r /Run -a
   /Run/HAS21
```

Line 1 runs docker in interactive mode.
Lines 2–4 mount the local directory onto the docker virtual environment.
Line 5 is the command to run HiSeq Analysis Software.

If you log out of your terminal session, the HiSeq Analysis Software v2.1 command can be prematurely terminated.

Use the following command line options to customize the workflow's input or output.

Option	Description
`-a`	The path to the root analysis folder. Use the same path for all analyses.
`-r`	The path to the run folder.
`-l`	[Optional] Localscratch directory for faster performance.

Demultiplexing

If the sample sheet defines multiple samples within the same lane, demultiplexing assigns clusters to the samples. For runs with Index Reads, demultiplexing compares each Index Read sequence to the index sequences specified in the sample sheet.

Demultiplexing separates data from pooled samples based on short index sequences that tag samples from different libraries. Index Reads are identified using the following steps:

- Samples are numbered starting from 1, based on the order they are listed in the sample sheet.
- Sample number 0 is reserved for clusters that were not successfully assigned to a sample.
- Clusters are assigned to a sample when the index sequence matches exactly or there is up to a single mismatch per Index Read.

NOTE

Illumina indexes are designed so that any index pair differs by ≥ 3 bases, allowing for a single mismatch in index recognition. Index sets that are not from Illumina can include pairs of indexes that differ by < 3 bases. In such cases, the software detects the insufficient difference and modifies the default index recognition (mismatch=1). Instead, the software performs demultiplexing using only perfect index matches (mismatch=0).

When demultiplexing is complete, the DemultiplexSummaryF1L1.txt file summarizes the following information:

- In the file name, **F1** represents the flow cell number.
- In the file name, **L1** represents the lane number.
- Reports demultiplexing results in a table with 1 row per tile and 1 column per sample, including sample 0.
- Reports the most commonly occurring sequences for the Index Reads.

The demultiplexing summary file is written to the QC folder. For the folder location, see *FASTQ Folder Structure* on page 7.

If the index sequence specified in the sample sheet is shorter than the actual Index Read, then the first cycles of the Index Read are used for demultiplexing. If there are no index sequences specified for Index or Index2, then the first or second Index Reads are not used for demultiplexing.

Make sure that all samples in the same lane have a compatible index scheme. Use the same number of cycles for Index and Index2.

FASTQ Folder Structure

- **[User-Specified FASTQ Folder]**
  - WorkflowError.txt
  - WorkflowLog.txt
  - **Fastq—Contains FASTQ files, reports, and stats**
    - **FASTQ/Reports/html**—Contains the primary metrics of FASTQ generation
    - **FASTQ/Stats**—Contains statistics from FASTQ generation
      - AdapterTrimming.txt
      - ConversionStats.xml
      - DemultiplexingStats.xml
      - DemuxSummaryF1L1.txt
      - FASTQSummaryF1L1.txt
    - **FASTQ/Undetermined S0**—Contains reads that could not be demultiplexed, whose index sequence(s) do not match any sample from the sample sheet
      - samplename_s1_L001_R1_001.fastq.gz

FASTQ File Generation

HiSeq Analysis Software v2.1 generates intermediate analysis files in the FASTQ format, which is a text format used to represent sequences and their quality scores. FASTQ files contain reads for each sample and their quality scores, excluding reads identified as inline controls and clusters that did not pass filter.

FASTQ file generation includes all tiles by default. If individual .bcl files or the associated .filter and .loc files are missing or corrupted, make sure that the ConvertMissingBCLsToNoCalls setting is set to true (value=1) in the sample sheet. This setting is true by default.

FASTQ File Format

FASTQ file is a text-based file format that contains base calls and quality values per read. Each record contains 4 lines:

- Identifier
- Sequence
- Plus sign (+)
- Phred quality scores in an ASCII +33 encoded format

The identifier is formatted as **@Instrument:RunID:FlowCellID:Lane:Tile:X:Y ReadNum:FilterFlag:0:SampleNumber** as shown in the following example:

```
@SIM:1:FCX:1:15:6329:1045 1:N:0:2
TCGCACTCAACGCCCTGCATATGACAAGACAGAATC
+
<>;##=><9=AAAAAAAAAA9#:<#<;<<<????#=
```

The X and Y coordinates in the read name are transformed relative to the raw pixel values according to the following calculation:

```
(int)((PixelX*10) + 1000 +0.5)
```

The instrument name is derived from the <Instrument> tag in the RunInfo.xml file. The run ID is derived from the Number attribute from the <Run ID> tag, or from the ID attribute. The following is an example of the tag, with the run number bolded:

```
<Run Id="11-07-23_BoltM11_0108_A-FCA012D_1" Number="108">
```

FASTQ File Names

FASTQ files are named with the sample name and the sample number. The sample number is a numeric assignment based on the order that the sample is listed in the sample sheet. For example:

Data\Intensities\BaseCalls\samplename_S1_L001_R1_001.fastq.gz

- **samplename**—The sample name provided in the sample sheet. If a sample name is not provided, the file name includes the sample ID.
- **S1**—The sample number, based on the order that samples are listed in the sample sheet, starting with 1. In this example, S1 indicates that this sample is the first sample listed in the sample sheet.

  NOTE

  Reads that cannot be assigned to any sample are written to a FASTQ file for sample number 0, and excluded from downstream analysis.

- **L001**—The lane number.
- **R1**—The read. In this example, R1 means Read 1. For a paired-end run, a file from Read 2 includes R2 in the file name.
- **001**—The last segment is always 001.

FASTQ files are compressed in the GNU zip format, as indicated by *.gz in the file name. FASTQ files can be uncompressed using tools such as gzip (command-line) or 7-zip (GUI).

QC Files and Other Output Files

To perform QC, first use the Sequencing Analysis Viewer (SAV) to view important quality metrics. If the generated FASTQ files are empty or nearly empty, view the output files for possible errors.

If the run does not include indexes, view the following files.

File	Description
WorkflowError.txt WorkflowLog.txt	Processing logs that contain additional information on the run.
InterOp files in the InterOp folder	Contain the %Q30 scores and additional information for QC.
FastqSummaryF1L*.txt	Contains the number of raw and PF clusters for each lane and tile on a per sample basis.

If the run includes indexes, view the following files in addition to the files listed above.

File	Description
IndexMetricsOut.bin	SAV uses this file with the other files in the InterOp folder.
DemultiplexSummaryF1L1.txt	Contains the percentage of each index sample for each lane and tile. This file also lists the most frequent indexes (unique or combination) found in the lanes. View this file when a sample has too many reads or no reads.
AdapterTrimming.txt	Lists the number of trimmed bases and percentage of bases for each tile. This file is present only if adapter trimming was specified for the run.
Undetermined_S0_L00*_R*_001.gz	Contains the reads with unidentifiable index reads. This file is present only when demultiplexing.

The following files are also output from the GenerateFASTQ workflow.

File	Description
CompletedJobInfo.xml	Written after analysis is complete. Contains information about the run, such as date, flow cell ID, software version, and other parameters.
ConversionStats.xml	Conversion statistics per tile, including raw cluster count, read number, yield Q30, yield, and quality score. Also conversion statistics per lane, including lane number.
DemultiplexingStatx.xml	Demultiplexing statistics per lane, barcode, sample, project, and flow cell.
SampleSheetUsed.csv	Sample sheet copied from the run folder

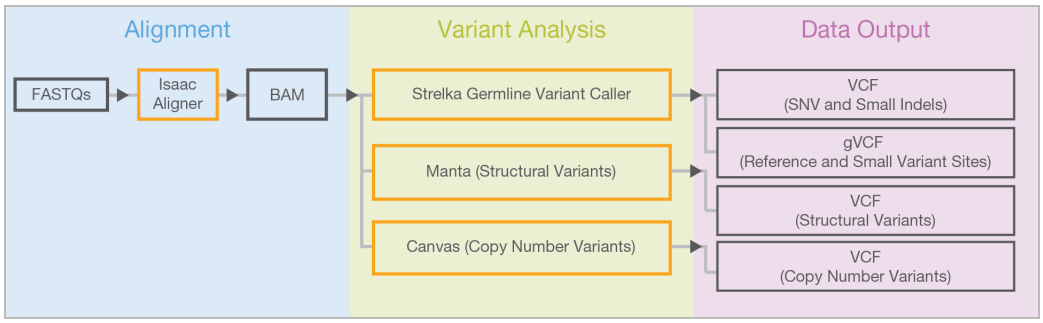
Resequencing Analysis Workflow

The Resequencing analysis workflow uses the Isaac Aligner and Strelka Germline Variant Caller to compare the DNA sequence in the sample against the human reference genome hg19. It identifies any small variants (SNPs or indels) and large structural variants (SVs and CNVs) relative to the reference sequence.

The Resequencing workflow uses the run folder as input.

The main output files are BAM files (containing the reads after alignment), VCF files (containing the variant calls), Genome VCF files (describing the calls for all variant and nonvariant sites in the genome), and SV VCF files (describing the calls for structural variants and copy number variants in the genome).

Figure 1 Resequencing Analysis Workflow



Run Resequencing Analysis

You can run the Resequencing analysis workflow from the run folder, FASTQ files, or BAM files.

From Run Folder

1 Identify a run folder for the analysis, which can be the original run folder for the flow cell.

2 In the top level of the run folder, create a sample sheet named SampleSheet.csv, like the following example. Make sure that the specified workflow and the GenomeFolder location are correct.
For more information, see *Sample Sheet* on page 20.

```
[Header],,,,,,,,
IEMFileVersion,4,,,,,,,
Investigator Name,S. w.,,,,,,,
Experiment Name,,,,,,,,
Date,2/12/2015,,,,,,,
Workflow,Resequencing,,,,,,,
Application,HiSeq FASTQ Only,,,,,,,
Assay,TruSeq HT,,,,,,,
Description,HiSeqX run,,,,,,,
Chemistry,Default,,,,,,,
,,,,,,,,
[Reads],,,,,,,,
151,,,,,,,,
151,,,,,,,,
,,,,,,,,
[Settings],,,,,,,,
ReverseComplement,0,,,,,,,
Adapter,AGATCGGAAGAGCACACGTCTGAACTCCAGTCA,,,,,,,
AdapterRead2,AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT,,,,,,,
,,,,,,,,
[Data],,,,,,,,
Lane,SampleID,SampleName,Sample_Plate,Sample_Well,I7_Index_
   ID,index,GenomeFolder,Sample_Project,Description
8,NA12878_GiaB1_ID,NA12878_GiaB1_Name,,,,,/genome/Homo_
   sapiens/UCSC/hg19/Sequence/WholeGenomeFasta,Project,
```

3 Start the Resequencing workflow using the same command used for the GenerateFASTQ. Example:

```
sudo docker run -i -t \
-v /etc/localtime:/etc/localtime:ro \
-v /Runs/150724_E00315_0078_BH5NWKCCXX /:/Run:rw \
-v /hg19-NSv4.2/:/genome \
illumina/isis:2.6.53.23 mono /opt/illumina/Isis/Isis.exe -r /Run -a
   /Run/HAS21
```

If available, fast localscratch is recommended. Use the following example command if using localscratch.

```
sudo docker run -i -t \
-v /etc/localtime:/etc/localtime:ro \
-v /mnt/localscratch:/localscratch:rw ]\
-v /Runs/150724_E00315_0078_BH5NWKCCXX/:/Run:rw ]\
-v /hg19-NSv4.2/:/genome
/illumina/isis:2.6.53.23 mono /opt/illumina/Isis/Isis.exe \
-r /Run -a /Run/HAS21 -l /localscratch
```

From FASTQ Files

1 Make sure that the FASTQ files meet the following criteria:
   - All FASTQ files are gzipped and have the file extension *.fastq.gz.
   - FASTQ filenames contain the string *_R1.fastq.gz or *_R1_*.fastq.gz for read 1. For paired-end samples, the matching read 2 files contain the same strings with R2 instead of R1. Lowercase r is also acceptable.
   - Only the last occurrence of a matching _R1_ or _R2_ in the file name is used. For example:
     - sample_r1_R1_fastq.gz and sample_r1_R2_fastq.gz works and is treated as a paired-end dataset.
     - sample_r1_R2_fastq.gz and sample_r2_R2_fastq.gz does ***not*** work and gives an error about a missing read 1 FASTQ file to match the read 2 FASTQ file.
     - sample_r1_R1_fastq.gz and sample_r2_R1_fastq.gz works, but is treated as 2 separate read 1 FASTQ files for a single-end dataset rather than a paired-end dataset.

2 In the top level of the run folder, create a sample sheet named SampleSheet.csv.
   - In the Settings section, add `FastqInput,UseExisting`
   - In the Data section, add `FastqFolder` as another column

   Example:

```
[Header]
Workflow,Resequencing
[Settings]
TranscriptSource,refseq
RunSVDetection,1
RunCNVDetection,1
FastqInput,UseExisting
[Data]
SampleID,SampleName,GenomeFolder,Lanes,PloidyX,PloidyY,FastqFolder
NA12878_micro,NA12878_micro, /genome/Homo_
   sapiens/UCSC/hg19/Sequence/WholeGenomeFasta,1,2,0, /Runs/150724_E00315_
   0078_BH5NWKCCXX/bcl2fastq_out
```

3 Start the Resequencing workflow using the same command used for the GenerateFASTQ. Example:

```
sudo docker run -i -t \
-v /etc/localtime:/etc/localtime:ro \
-v /Runs/150724_E00315_0078_BH5NWKCCXX /:/Run:rw \
```

```
-v /hg19-NSv4.2/:/genome \
illumina/isis:2.6.53.23 mono /opt/illumina/Isis/Isis.exe -r /Run -a
   /Run/HAS21
```

From BAM files

1 In the top level of the run folder, create a sample sheet named SampleSheet.csv.
In the [Data] section, add `InputBam` as another column. Example:

```
[Header]
Workflow,Resequencing
[Settings]
TranscriptSource,refseq
SVAnnotation,Nirvana
RunSVDetection,1
RunCNVDetection,1
FastqInput,0
[Data]
SampleID,SampleName,GenomeFolder,Lanes,PloidyX,PloidyY,InputBam
NA12878_micro,NA12878_micro, /genome/Homo_
   sapiens/UCSC/hg19/Sequence/WholeGenomeFasta,1,2,0, /Runs/150724_E00315_
   0078_BH5NWKCCXX/bam
```

2 Start the Resequencing workflow using the same command used for the GenerateFASTQ. Example:

```
sudo docker run -i -t \
-v /etc/localtime:/etc/localtime:ro \
-v /Runs/150724_E00315_0078_BH5NWKCCXX /:/Run:rw \
-v /hg19-NSv4.2/:/genome \
illumina/isis:2.6.53.23 mono /opt/illumina/Isis/Isis.exe -r /Run -a
   /Run/HAS21
```

Resequencing Analysis Folder Structure

[User-Specified Analysis Folder]

- samplename_S1.bam
- samplename_S1.bam.bai
- samplename_S1.bam.md5sum
- samplename_S1.CNV.CoverageAndVariantFrequency.txt
- samplename_S1.CoverageHistogram.txt
- samplename_S1.ExonCoverageTable.txt
- samplename_S1.genome.vcf.gz
- samplename_S1.genome.vcf.gz.tbi
- samplename_S1.json.gz
- samplename_S1.report.html
- samplename_S1.report.pdf
- samplename_S1.ROH.bed
- samplename_S1.ROH.txt
- samplename_S1.summary.csv
- samplename_S1.png
- samplename_S1.SV.json.gz
- samplename_S1.vcf.gz
- samplename_S1.vcf.gz.tbi
- SampleSheetUsed.csv
- Summary.htm
- Summary.xml

Checkpoints—Contains *.json files of all analysis steps.

Fastq—Empty, unless generating FASTQ files as part of the workflow.

Logging—Contents are used for troubleshooting only.

Resequencing Workflow Output Files

File Name	Description
*.bam	Contains aligned and unaligned reads sorted by mapping position to the reference sequence.
*.bam.bai	BAM index used to quickly query regions of aligned reads from the BAM file.
*.bam.md5sum	Text file containing MD5 checksum of the BAM file.
*.CNV.CoverageAnd VariantFrequency.txt	Table reporting the relative coverage, variant allele frequency histogram, and copy number call for the genome broken down into 1 entry per (by default) 100 kilobases.

File Name	Description
*.CoverageHistogram.txt	Contains coverage depth information for every chromosome. This file can be used to generate a coverage histogram.
*.ExonCoverageTable.txt	Tab-delimited file providing chromosome, start, stop, name, median coverage, and percent call ability for all exons.
*.genome.vcf.gz	gVCF file containing variant and small variant calls. A tabix index file is also produced.
*.json.gz	Extended annotation data for the SNVs and indels.
*.report.pdf / *.report.html	Summary report for the sample. Provides several high-level statistics pulled from *.summary.csv, such as genome coverage and % aligned.
*.ROH.bed	A .bed file of the ROH calls. Contains extra colums (4) ROH score, (5) number of unfiltered homozygous alternate calls in the ROH, and (6) number of unfiltered heterozygous SNV calls in the ROH.
*.ROH.txt	A text file reporting overall metrics of the ROH caller. Reported metrics are percent SNVs in ROH, unfiltered het/hom, filtered het/hom (large ROH), percent SNVs in large ROH, and number of large ROH.
*.summary.csv	Summary statistics for the sample, for parsing by downstream tools.
*.summary.png	A genome-wide view of b-allele frequencies, and structural variants.
*.SV.vcf.gz	VCF file containing SV and CNV calls. Also includes reference calls from the CNV caller. A *.json and a tabix index file also gets produced.
*.vcf.gz	VCF file containing variant calls. A tabix index file is also produced.
SampleSheetUsed.csv	Sample Sheet used for the analysis.
Summary.htm	HTML file containing statistics on the flow cell on a per tile basis.
Summary.xml	XML version of the summary file.

Resequencing Analysis Report

The Resequencing workflow outputs a PDF and an HTML report that provides an overview of statistics per sample.

Sample Information

The Sample Information table provides statistics about the sample and alignment quality.

Statistic	Definition
Total PF Reads	The total number of reads passing filter.
% Q30 Bases	The percentage of bases with a quality score of 30 or higher.
Total Aligned Reads 1 and 2	The total number of aligned reads for reads 1 and 2.
% Aligned Reads 1 and 2	The percentage of aligned passing filter reads for reads 1 and 2.
Total Aligned Bases Reads 1 and 2	The total number of aligned bases for reads 1 and 2.
% Aligned Bases Reads 1 and 2	The percentage of aligned bases for reads 1 and 2.
Mismatch Rate Reads 1 and 2	The average percentage of mismatches of reads 1 and 2 over all cycles.

Coverage Histogram

The Coverage histogram shows the number of reference bases plotted against the depth of coverage.

The report also includes the mean coverage, which is the total number of aligned bases divided by the genome size. The Coverage histogram considers duplicates, but the mean coverage does not.

Small Variants Summary

The Small Variants summary table provides metrics about the number of SNVs, insertions, and deletions.

Statistic	Definition
Total Passing	The total number of variants present in the data set that pass the variant quality filters.
Het/Hom Ratio	The number of heterozygous variants divided by the number of homozygous variants.
Ts/Tv Ratio	Transition rate of SNVs that pass the quality filters divided by the transversion rate of SNVs that pass the quality filters. Transversions are interchanges between purine and pyrimidine bases (for example, A to T).

Structural Variants Summary

The Structural Variants summary table separates the structural variant output into the classes of called variants. The table also includes the total number of structural variants and the overlap with annotated genes. All counts are based on PASS filter variants.

Fragment Length Summary

The Fragment Length summary table provides statistics on the sequenced fragments.

Statistic	Definition
Fragment Length Median	The median length of the sequenced fragment. The fragment length is based on the locations where a read pair aligns to the reference. The read mapping information is parsed from the BAM files.
Minimum	The minimum length of the sequenced fragment.
Maximum	The maximum length of the sequenced fragment.
Standard Deviation	The standard deviation of the sequenced fragment length.

Duplicate Information

The Duplicate Information table provides the percentage of paired reads that have duplicates.

Analysis Details

The Analysis Details tables provide the settings and software packages used for the analysis.

Tumor-Normal Analysis Workflow

HiSeq Analysis Software v2.1 includes a Tumor-Normal analysis workflow that harnesses a suite of proven algorithms that are optimized for the complexities of tumor samples. The software delivers a set of accurate somatic variants when compared with a matched normal sample.

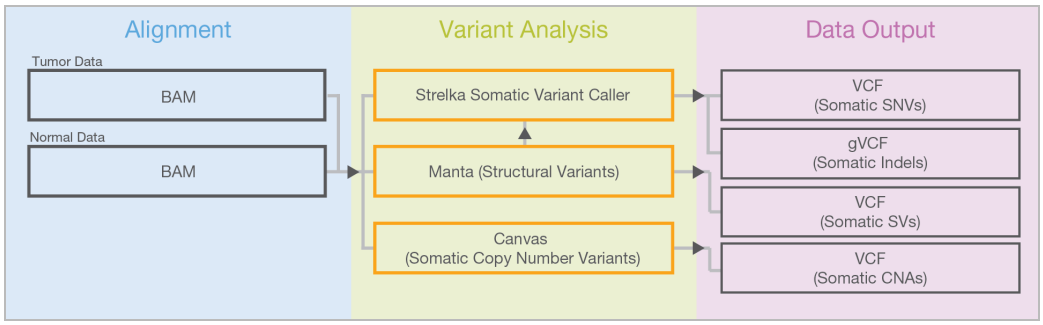
Following alignment of both the tumor and normal sample, the Tumor-Normal workflow is used to identify the somatic variants (SNVs, small indels, and structural variants) that are unique to the tumor sample. When analyzing the tumor sample, use the Alignment analysis workflow, rather than the Resequencing analysis workflow, for the alignment step. The alignment analysis workflow is identical to the resequencing analysis workflow, but does not include the variant call

analysis. The variant call analysis used in the Resequencing analysis workflow is not valid for a tumor sample, which can contain somatic variants. The variant calling methods used in the Resequencing analysis workflow assume a diploid genotype.

For optimal results, Illumina recommends a minimum coverage of 30x for the normal sample and 60x coverage for the tumor sample.

The following figure outlines the Tumor-Normal analysis workflow and generated files. The Tumor-Normal analysis workflow uses 3 interconnected somatic variant callers—Strelka Somatic Variant Caller for SNVs and small somatic variants, Manta for large indel and structural variants, and Canvas for somatic copy number variants.

Figure 2 Tumor-Normal Analysis Workflow



Run Tumor-Normal Analysis

1 Run the Resequencing workflow for each tumor and normal sample. A sample can span multiple flow cells. For instructions, see *Resequencing Analysis Workflow* on page 9.
To turn off the analyses that are not meaningful for tumor samples, add the following to the [Settings] section of the sample sheet.
- `VariantCaller,None`
- `RunSVDetection,0`
- `RunCNVDetection,0`

2 Run the Tumor-Normal workflow for each tumor-normal pair.
- Create a sample sheet named SampleSheet.csv like the following example.
```
[Header],,,
Investigator Name,Billy,,
Project Name,My First Tumor Normal Analysis,,
Date,2/1/2014,,
Workflow,TumorNormal,,
[Settings]
[Data]
SampleID,SampleName,MatchedNormalID,AlignmentPath
Normal1,HCC2218BL,,/Run/HAS21_Resequencing/NA12878_GiaB1_Name_S1.bam
Tumor1,HCC2218C, Normal1,/Run/HAS21_Tumor/NA12878_GiaB1_Name_S1.bam
```
- Place the sample sheet in the location specified in the command. Command example:
```
sudo docker run -i -t \
-v /etc/localtime:/etc/localtime:ro --privileged
-v /mnt/150724_E00315_0078_BH5NWKCCXX/:/Run:rw
-v /hg19-NSv4.2/:/genome illumina/isis:2.6.53.23 mono
   /opt/illumina/Isis/Isis.exe
```

```
-r /Run/HAS21_TumorNormalSampleSheet
-a /Run/HAS21_TumorNormal
```

The command assumes that the sample sheet is placed in /mnt/150724_E00315_0078_BH5NWKCCXX/HAS21_TumorNormalSampleSheet.

The command also specifies that the location for the analysis output is /mnt/150724_E00315_0078_BH5NWKCCXX/HAS21_TumorNormal.

Use the following command line options to customize the run.

Option	Description
`-l`	[Optional] The path to the local storage. If specified, the input files for the specified workflow are copied to the local storage of the compute node (local hard drive). If there is available storage on the local node, the `-l` option is recommended as standard practice.

Tumor-Normal Analysis Folder Structure

[User-Specified Analysis Folder]

- Checkpoint.txt
- CompletedJobInfo.xml
- normalsample_tumorsample_G1_P1.somatic.json.gz
- normalsamplename_tumorsamplename _G1_P1.somatic.json.gz
- normalsamplename_tumorsamplename _G1_P1.somatic.SV.vcf.gz
- normalsamplename_tumorsamplename _G1_P1.somatic.SV.vcf.gz.bi
- normalsamplename_tumorsamplename _G1_P1.somatic.vcf.gz
- normalsamplename_tumorsamplename _G1_P1.somatic.vcf.gz.tbi
- ploidy.bed.gz
- ploidy.bed.gz.tbi
- SampleSheetUsed.csv
- WorkflowError.txt
- WorkflowLog.txt

Logging Temp — Used for troubleshooting only.

CNV_G1_P1 — Used for troubleshooting only.

Tumor-Normal Workflow Output Files

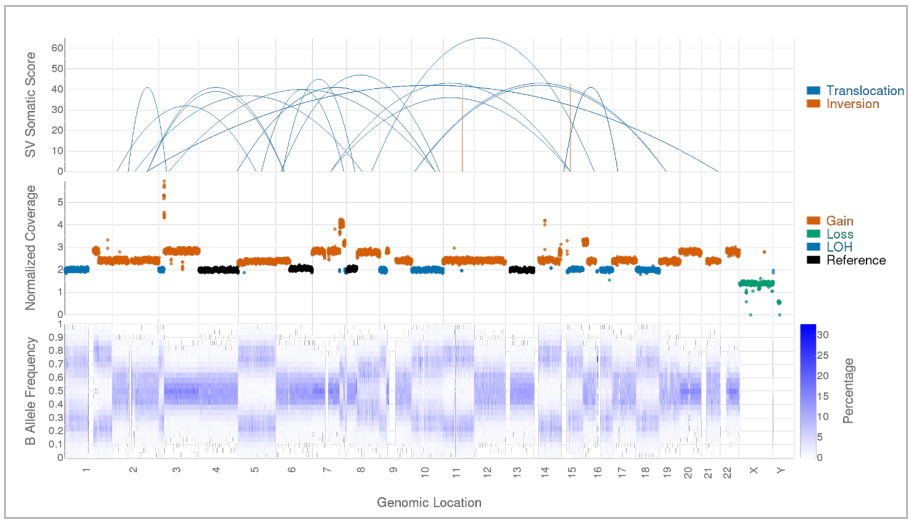
File	Description
*.somatic.vcf.gz	VCF containing somatic SNVs and small indels identified by Strelka. A tabix index file is also produced.
*.somatic.json.gz	Extended annotation data for the somatic SNVs and indels.
*.somatic.SV.vcf.tz	VCF containing somatic SVs and CNVs identified by Manta and Canvas, respectively. Estimated Ploidy and purity are included in the VCF header. A tabix index file is also produced.
*.summary.png	A genome-wide view of coverage levels, b-allele frequencies, and structural variants.

File	Description
*.report.pdf	A high-level sample report including important metrics, plot of the coverage histogram, settings used, and tool versions.
*.summary.csv	List of all available metrics for the normal and tumor samples. For more information, see *Summary Metrics* on page 18.

Somatic Analysis Report

The Tumor-Normal workflow outputs a PDF report that provides a summary of the somatic analysis results. The report includes a high-level summary of the tumor along with normal samples that include the number of reads, percent aligned bases, and a histogram of genome coverage. The report summarizes the following:

- Somatic variant calls (number of events by type and sequence context)
- A plot showing:
  - SVs
  - coverage
  - b allele frequency
- Software version used for the analysis



Summary Metrics

Applicable summary metrics from the Resequencing workflow are also included in the somatic Summary.csv output file. The resequencing metrics are prefixed with either Normal or Tumor to indicate the associated sample.

Metric	Definition	Source
Estimated Chromosome Count	Estimated total number of chromosomes in the tumor cells (sum of the average copy number across all autosomes and allosomes). This number is 46 for a XX or XY sample with a wild-type copy number. This number can be significantly larger for genomes with many amplifications (eg, triploid and tetraploid chromosomes) and smaller for largely haploid tumor samples.	SV VCF header (##EstimatedChromosomeCount line)
Estimated Purity	Estimated tumor purity, from 0 to 1.	SV VCF header (##EstimatedTumorPurity line)

Metric	Definition	Source
Mean Somatic SNV Frequency	Mean variant allele frequency for passing filter SNV calls. Variant allele frequency is the ratio of A/(A+B), where A is the number of reads (after filtering) with the variant allele, and B is the number of reads (after filtering) with the reference allele. (In the VCF file, the tags AU/CU/GU/TU give the depth for each base, and the REF and ALT columns, respectively, give the 2 bases of interest.)	Somatic SNV/Indel VCF file
Median Somatic SNV Distance	Median distance, in base pairs, between pairs of consecutive passing filter SNV calls on the same chromosome.	Somatic SNV/Indel VCF file
Overall Ploidy	Mean copy number across all bases of the autosomes and allosomes; roughly equal to 2 for wild-type data.	SV VCF header (##OverallPloidy line)
Somatic CNV	Somatic CNV metrics.	Somatic SV VCF file
Somatic Indels	Number of passing filter indel calls.	Somatic SNV/Indel VCF file
Somatic SNVs	Number of PF SNV calls.	Somatic SNV/Indel VCF file
Somatic SNVs (Percent found in dbSNP)	Percentage of PF SNV calls found in dbSNP (has a nonempty ID column after annotation).	Somatic SNV/Indel VCF file
Somatic SNVs, Insertions, Deletions, Indels	Somatic SNV metrics.	somatic.vcf.gz file
Somatic SV	Somatic SV metrics.	Somatic SV VCF file
Tumor Sample ID	Tumor sample ID.	Sample Sheet
Tumor Sample Name	Tumor sample name.	Sample ID

Input Files

The following file is user-provided.

- **SampleSheet.csv**—Contains user-specified analysis options for the run, including SampleIDs and index sequences. Create the SampleSheet.csv file for the GenerateFASTQ step and save it in the top level of the run folder.

HiSeq X generates the following files. These files are found in the run folder after a sequencing run.

- **RunInfo.xml**—Produced by RTA and contains information about cycles per read, etc.
- Base call files (**s_x_yyyy.bcl.gz**)—Include base calls for lane x, tile yyyy, cycle z. Located in Data\Intensities\BaseCalls\L00x\Cz.1.
- Filter files (**s_x_yyyy.filter**)—Include filter flags for lane x, tile yyyy. Located in Data\Intensities\BaseCalls\L00x\.
- Position files (**s.locs**)—Include locations for all lanes. Located in Data\Intensities.

Sample Sheet

The sample sheet identifies the samples included in an analysis and contains the index information used in sample demultiplexing. To customize the sample sheet for the GenerateFASTQ workflow, enter the following parameters in the sample sheet.

Settings Section

Each line in the Settings section contains a parameter name in the first column and a value in the second column. Settings are not case-sensitive.

Parameter	Description
Adapter	Specify the 5' portion of the adapter sequence to prevent reporting sequence beyond the sample DNA. To trim 2 or more adapters, separate the sequences by a plus (+) sign. (Example: Nextera Mate Pair Libraries – CTGTCTCTTATACACATCT+AGATGTGTATAAGAGACAG). Specific to GenerateFASTQ.
AdapterRead2	Specify the 5' portion of the Read 2 adapter sequence to prevent reporting sequence beyond the sample DNA. Use this setting to specify a different adapter than the 1 specified in the Adapter setting. If not specified, the Adapter setting is used for Read 2. Specific to GenerateFASTQ.
RunHLATyping	Specify the HLA typer that identifies the most likely genotypes for 6 HLA genes. Disabled by default. Specific to the Resequencing workflow.
RunExpansionHunter	Specify the calling of triplet repeat expansion in the Resequencing workflow. Disabled by default.

Data Section

The following table includes the parameters and descriptions of the Sample Sheet's Data fields.

Parameter	Description
Lane	[Optional] Lane numbers used for a sample. If not specified, all lanes are used. Each lane can be entered as a separate row, delimited by +. For example: 1+2+3. Specific to the GenerateFASTQ workflow.
SampleID	Sample ID. Combined with SampleProject to make a unique identifier for the sample. Specified in all workflows.
SampleName	[Optional] Sample Name. If this field is completed, the sample name is used when naming FASTQ files.
Index	[Optional] Index bases used to identify a sample. Specific to the GenerateFASTQ workflow.
Index2	[Optional] Second index bases used to identify a sample. Specific to the GenerateFASTQ workflow.
GenomeFolder	Location of genome used for alignment. Varies depending on where the folder was extracted during setup. Specific to the Resequencing workflow.
PloidyX	[Optional, integer] X chromosome count for this sample. Used to configure ploidy correction in the Resequencing workflow.
PloidyY	[Optional, integer] Y chromosome count for this sample. Used to configure ploidy correction in the Resequencing workflow.
InputBam	[Optional] Location of BAM files. Used to run the Resequencing workflow from the BAM files.
FastqFolder	[Optional] Location of FASTQ files. Used to run the Resequencing workflow from the FASTQ folder.

Methods for the Resequencing Workflow

This section describes the methodologies for the Resequencing and Alignment algorithm in HiSeq Analysis Software v2.1. The Resequencing analysis and Alignment analysis workflows are identical, except that the Alignment analysis workflow does not contain variant calling and is only recommended for tumor samples.

Overview

After the sequencer generates base calls and quality scores, the resulting data get aligned to the reference genome. Next, assembly and variant calling occurs.

Alignment and variant calling happen within Isaac Aligner, Starling, Canvas, and Manta, producing the following output:

- Realigned and duplicate marked reads in a BAM file format.
- Variants in a VCF file format.
- An additional Genome VCF (gVCF) file. This file features an entry for every base in the reference, which differentiates reference calls and no calls, and a summary of quality. The reference calls are block compressed and all single nucleotide polymorphisms and indels are included. Structural Variants and CNVs are contained in separate files.

Genome Specific Details

Illumina uses iGenomes as the source for the reference genomes used in alignment and assembly. The internally used genome can differ slightly from the iGenomes version in that the pseudoautosomal region (PAR) of the Y chromosome is hard-masked with Ns. Use iGenomes to avoid false mapping of reads; any mapping occurring in the PAR regions maps to the X chromosome. The reference genome used for a particular build is specified in both the BAM header and the summary output files.

The ncbi37/hg18/GRCh37 PAR regions are defined as follows.

Name	Chr	Start	Stop
PAR#1	X	60,001	2,699,520
PAR#2	X	154,931,044	155,260,560
PAR#1	Y	10,001	2,649,520
PAR#2	Y	59,034,050	59,363,566

The ncbi38/hg38/GRCh38 PAR regions are defined as follows.

Name	Chr	Start	Stop
PAR#1	X	10,001	2,781,479
PAR#2	X	155,701,383	156,030,895
PAR#1	Y	10,001	2,781,479
PAR#2	Y	56,887,903	57,217,415

Isaac Aligner

The Isaac software uses the following steps to align DNA sequencing data with read lengths of 32–150 bp, single or paired-end, and low error rates.

- **Candidate mapping positions**—Identifies the complete set of relevant candidate mapping positions using a 32-mer seed-based search.
- **Mapping selection**—Selects the best mapping among all candidates.
- **Alignment score**—Determines alignment scores for the selected candidates based on a Bayesian model.
- **Alignment output**—Generates final output in a sorted and duplicate-marked BAM file, realigned indels, and a summary file.

Candidate Mapping

To align reads, Isaac Alignment first identifies a small but complete set of relevant candidate mapping positions. The aligner begins with a seed-based search using 32-mers from the extremities of the read as seeds. It then performs another search using different seeds for only those reads that were already mapped ambiguously with the first-pass seeds.

Mapping Selection

Following a seed-based search, Isaac Alignment selects the best mapping among all candidates.

For paired-end data sets, mappings where only 1 end is aligned (called orphan mappings) prompt a local search to find additional mapping candidates. These candidates are called shadow mappings. They are defined through the expected minimum and maximum insert size. After optional trimming of low quality 3′ ends and adapter sequences, the possible mapping positions of each fragment are compared. This step accounts for any available pair-end information, possible gaps using a banded Smith-Waterman gap aligner, and possible shadows. The selection is based on the Smith-Waterman score and on the log-probability of each mapping.

Alignment Scores

The alignment scores of each read pair are based on a Bayesian model, where the probability of each mapping is inferred from the base qualities and the positions of the mismatches. The final mapping quality (MAPQ) is the alignment score, truncated to 60 for scores above 60, and corrected based on known ambiguities in the reference flagged during candidate mapping. Following alignment, reads are sorted. Further analysis is performed to identify duplicates and, optionally, to realign indels.

Alignment Output

After sorting the reads, Isaac Alignment generates compressed binary alignment output files, called BAM (*.bam) files, using the following process:

- **Marking duplicates**—Detection of duplicates is based on the location and observed length of each fragment. Isaac Alignment identifies and marks duplicates even when they appear on oversized fragments or chimeric fragments.
- **Realigning indels**—Isaac Alignment tracks previously detected indels, over a window large enough for the current read length, and applies the known indels to all reads with mismatches.
- **Generating BAM files**—The first step in BAM file generation is creation of the BAM record, which contains all required information except the name of the read. Isaac Alignment reads data from the FASTQ header for each read to generate the read names. Data are then compressed into blocks of 64 kb or less to create the BAM file.

Strelka Germline Variant Caller

Strelka identifies single nucleotide variants (SNVs) and small indels using the following steps:

- **Read filtering**—Filters out reads failing quality checks.

- **Indel calling** — Identifies a set of possible indel candidates and realigns all reads overlapping the candidates using a multiple sequence aligner.
- **SNV calling** — Computes the probability of each possible genotype given the aligned read data and a prior distribution of variation in the genome.
- **Indel genotypes** — Calls indel genotypes and assigns probabilities.
- **Variant call output** — Generates output in a compressed genome variant call (gVCF) file. See *gVCF Files* on page 1 for details.

Indel Candidates

Input reads are filtered by removing any of the following reads:

- Reads that failed base calling quality checks.
- Reads marked as PCR duplicates.
- Paired-end reads not marked as a proper pair.
- Reads with a mapping quality < 20.

Indel Calling

The variant caller proceeds with candidate indel discovery and generates alternate read alignments based on candidate indels. During realignment, the variant caller selects a representative alignment to use for site genotype calling and depth summarizing by the SNV caller.

SNV Calling

The variant caller filters the set of filtered and realigned reads for SNV calling without affecting indel calls. Any contiguous trailing sequence of N base calls is trimmed from the end of the read. Using a mismatch density filter, reads with an unexpectedly high number of disagreements with the reference are masked as follows.

- The variant caller identifies each insertion or deletion as a single mismatch.
- Base calls with more than 2 mismatches to the reference sequence within 20 bases of the call are ignored.
- If the call occurs within the first or last 20 bases of a read, the mismatch limit is applied to a 41-base window at the corresponding end of the read.
- The mismatch limit is applied to the entire read when the read length is 41 or shorter.

Indel Genotypes

The variant caller filters all bases marked by the mismatch density filter and any N base calls that remain after the end-trimming step. These filtered base calls are not used for site-genotyping, but they appear in the filtered base call counts in the variant caller output for each site.

All remaining base calls are used for site-genotyping. The genotyping method adjusts the joint error probability that is calculated from multiple observations of the same allele on each strand of the genome. This correction accounts for the possibility of error dependencies.

This method treats the highest-quality base call from each allele and strand as an independent observation and leaves the associated base call quality scores unmodified. Quality scores for subsequent base calls for each allele and strand are then adjusted. This adjustment increases the joint error probability of the given allele above the error expected from independent base call observations.

Variant Call Output

After the SNV and indel genotyping methods are complete, the variant caller applies a final set of heuristic filters to produce the final set of calls in the output.

The output in the genome variant call (gVCF) file captures the genotype at each position and the probability that the consensus call differs from reference. This score is expressed as a Phred-scaled quality score.

Repeat Expansion Calling (Expansion Hunter)

Expansion Hunter calls triplet repeat expansions. The list of loci called is specified in theAnnotation/ExpansionHunterDiseaseSpec folder for the reference genome.

This caller is disabled by default. To enable the caller, set RunExpansionHunter to true in the Settings section of the sample sheet.

HLA Typing

Enable the HLA typer to identify the most likely genotypes for 6 HLA genes. The output is a tab delimited text file named SampleID_S#.HLA.txt. The file includes 4 columns—GeneName, Allele1, Allele2, and Rank. This file lists the 10 most likely pairs of alleles for each of the 6 HLA genes.

HLA typing is disabled by default. To enable HLA typing, set RunHLATyping to true in the Settings section of the sample sheet.

gVCF (Genome VCF)

Human genome sequencing applications require sequencing information for both variant and nonvariant positions, yet there is no common exchange format for such data. gVCF addresses this issue.

gVCF is a set of conventions applied to the standard variant call format (VCF). These conventions allow representation of genotype, annotation, and additional information across all sites in the genome, in a reasonably compact format. Typical human whole-genome sequencing results expressed in gVCF with annotation are less than 1.7 GB, or about 1/50 the size of the BAM file used for variant calling.

gVCF is also equally appropriate for representing and compressing targeted sequencing results. Compression is achieved by joining contiguous nonvariant regions with similar properties into single block VCF records. To maximize the utility of gVCF, especially for high stringency applications, the properties of the compressed blocks are conservative. Block properties such as depth and genotype quality reflect the minimum of any site in the block. The gVCF file is also a valid VCF v4.1 file, and can be indexed and used with existing VCF tools such as tabix and IGV. This feature makes the file convenient both for direct interpretation and as a starting point for further analysis.

gvcftools

Illumina has created a full set of utilities aimed at creating and analyzing Genome VCF files. For information and downloads, visit the gvcftools website at sites.google.com/site/gvcftools/home.

Examples

The following is a segment of a VCF file following the gVCF conventions for representation of nonvariant sites and, more specifically, using gvcftools block compression and filtration levels.

In the following gVCF example, nonvariant regions are shown in normal text and variants are shown in **bold**.

NOTE

The variant lines can be extracted from a gVCF file to produce a conventional variant VCF file.

```
chr20 676337 . T . 0.00 PASS END=676401;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:143:51:0
chr20 676402 . A . 0.00 PASS END=676441;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:169:57:0
chr20 676442 . T G 287.00 PASS SNVSB=-30.5;SNVHPOL=3 GT:GQ:GQX:DP:DPF:AD
   0/1:316:287:66:1:33,33
chr20 676443 . T . 0.00 PASS END=676468;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:202:68:1
chr20 676469 . G . 0.00 PASS . GT:GQX:DP:DPF 0/0:199:67:5
chr20 676470 . A . 0.00 PASS END=676528;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:157:53:0
chr20 676529 . T . 0.00 PASS END=676566;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:120:41:0
chr20 676567 . C . 0.00 PASS END=676574;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:114:39:0
chr20 676575 . A T 555.00 PASS SNVSB=-50.0;SNVHPOL=3 GT:GQ:GQX:DP:DPF:AD
   1/1:114:114:39:0:0,39
chr20 676576 . T . 0.00 PASS END=676625;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:95:36:0
chr20 676626 . T . 0.00 PASS END=676650;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:117:40:0
chr20 676651 . T . 0.00 PASS END=676698;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:90:31:0
chr20 676699 . T . 0.00 PASS END=676728;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:69:24:0
chr20 676729 . C . 0.00 PASS END=676783;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:57:20:0
chr20 676784 . C . 0.00 PASS END=676803;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:51:18:0
chr20 676804 . G A 62.00 PASS SNVSB=-7.5;SNVHPOL=2 GT:GQ:GQX:DP:DPF:AD
   0/1:95:62:17:0:11,66
chr20 676805 . C . 0.00 PASS END=676818;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:48:17:0
chr20 676819 . T . 0.00 PASS END=676824;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:39:14:0
chr20 676825 . A . 0.00 PASS END=676836;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:30:11:0
chr20 676837 . T . 0.00 LowGQX END=676857;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:21:8:0
chr20 676858 . G . 0.00 PASS END=676873;BLOCKAVG_min30p3a GT:GQX:DP:DPF
   0/0:30:11:0
```

In addition to the nonvariant and variant regions in the example, there is also 1 nonvariant region from [676837,676857] that is filtered out due to insufficient confidence that the region is homozygous reference.

Conventions

Any VCF file following the gVCF convention combines information on variant calls (SNVs and small-indels) with genotype and read depth information for all nonvariant positions in the reference. Because this information is integrated into a single file, distinguishing variant, reference, and no-call states for any site of interest is straightforward.

The following subsections describe the general conventions followed in any gVCF file, and provide information on the specific parameters and filters used in the Isaac workflow gVCF output.

NOTE

gVCF conventions are written with the assumption that only one sample per file is being represented.

Interpretation

gVCFs file can be interpreted as follows:

- **Fast interpretation**—As a discrete classification of the genome into variant, reference, and no-call loci. This classification is the simplest way to use the gVCF. The Filter fields for the gVCF file have already been set to mark uncertain calls as filtered for both variant and nonvariant positions. Simple analysis can be performed to look for all loci with a filter value of PASS and treat them as called.
- **Research interpretation**—As a statistical genome. Additional fields, such as genotype quality, are provided for both variant and reference positions to allow the threshold between called and uncalled sites to be varied. These fields can also be used to apply more stringent criteria to a set of loci from an initial screen.

External Tools

gVCF is written to the VCF 4.1 specifications, so any tool that is compatible with the specification (such as IGV and tabix) can use the file. However, certain tools are not appropriate if they:

- Apply algorithms to VCF files that make sense for only variant calls (as opposed to variant and nonvariant regions in the full gVCF)
- Are only computationally feasible for variant calls

For these cases, extract the variant calls from the full gVCF file.

Special Handling for Indel Conflicts

Sites that are filled in inside deletions have additional treatment.

- **Heterozygous Deletions**—Sites inside heterozygous deletions have haploid genotype entries (ie, 0 instead of 0/0, 1 instead of 1/1). Heterozygous SNVs are marked with the SiteConflict filter and their original genotype is left unchanged. Sites inside heterozygous deletions cannot have a genotype quality score higher than the enclosing deletion genotype quality.
- **Homozygous Deletions**—Sites inside homozygous deletions have genotype set to period (.), and site and genotype quality are also set to period (.).
- **All Deletions**—Sites inside any deletion are marked with the filters of the deletion, and more filters can be added pertaining to the site itself. These modifications reflect the idea that the enclosing indel confidence bounds the site confidence.
- **Indel Conflicts**—In any region where overlapping deletion evidence cannot be resolved into 2 haplotypes, all indel and set records in the region are marked with the IndelConflict filter.

Table 1 Indel Conflict Filters

ID	Type	Description
IndelConflict	site/indel	Locus is in region with conflicting indel calls.
SiteConflict	site	Site genotype conflicts with proximal indel call. This conflict is typically a heterozygous genotype found inside a heterozygous deletion.

Representation of Nonvariant Segments

This section includes the following subsections:

- Block representation using END key
- Joining nonvariant sites into a single block record
- Block sample values
- Nonvariant block implementations

Block Representation Using END Key

Continuous nonvariant segments of the genome can be represented as single records in gVCF. These records use the standard END INFO key to indicate the extent of the record. Even though the record can span multiple bases, only the first base is provided in the REF field (to reduce file size). Following is a simplified example of a nonreference block record:

```
##INFO=<ID=END,Number=1,Type=Integer,Description="End position of the variant
   described in this record">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT NA19238
chr1 51845 . A . . PASS END=51862
```

The example record spans positions [51845,51862].

Joining Nonvariant Sites Into a Single Block Record

Address the following issues when joining adjacent nonvariant sites into block records:

- The criteria that allow adjacent sites to be joined into a single block record.
- The method to summarize the distribution of SAMPLE or INFO values from each site in the block record.

At any gVCF compression level, a set of sites can be joined into a block if they meet the following:

- Each site is nonvariant with the same genotype call. Expected nonvariant genotype calls are {0/0, 0, ./., .}.
- Each site has the same coverage state, where coverage state refers to whether at least 1 read maps to the site. For example, sites with 0 coverage cannot be joined into the same block with covered sites.
- Each site has the same set of FILTER tags.
- Sites have less than a threshold fraction of nonreference allele observations compared to all observed alleles (based on AD and DP field information). This threshold is used to keep sites with high ratios of nonreference alleles from being compressed into nonvariant blocks. In the Strelka Germline Variant Caller gVCF output, the maximum nonreference fraction is 0.2.

Block Sample Values

Any field provided for a block of sites, such as read depth (using the DP key), shows the minimum observed value among all sites encompassed by the block.

Nonvariant Block Implementations

Files conforming to the gVCF conventions delineated in this document can use different criteria for creation of block records, depending on the desired trade-off between compression and nonvariant site detail. The Strelka Germline Variant caller provides the blocking scheme min30p3a as the nonvariant block compression scheme.

Each sample value shown for the block, such as the depth (using the DP key), is restricted to have a range where the maximum value is within 30% or 3 of the minimum. Therefore, for sample value range [x,y], $y \leq x+\max(3, x*0.3)$. This range restriction applies to all sample values written in the final block record.

Genotype Quality for Variant and Nonvariant Sites

The gVCF file uses an adapted version of genotype quality for variant and nonvariant site filtration. This value is associated with the GQX key. The GQX value is intended to represent the minimum of Phred genotype quality (assuming the site is variant, assuming the sites is nonvariant).

You can use this value to allow a single value to be used as the primary quality filter for both variant and nonvariant sites. Filtering on this value corresponds to a conservative assumption appropriate for applications where reference genotype calls must be determined at the same stringency as variant genotypes, for example:

- An assertion that a site is homozygous reference at GQX ≥ 20 is made assuming the site is variant.
- An assertion that a site is a nonreference genotype at GQX ≥ 20 is made assuming the site is nonvariant.

Filter Criteria

The gVCF FILTER description is divided into 2 sections, the first describes filtering based on genotype quality while the second describes all other filters.

NOTE

These filters are default values used in the current Strelka Germline Variant Caller implementation. However, no set of filters or cutoff values are required for a file to conform to gVCF conventions.

The genotype quality is the primary filter for all sites in the genome. In particular, traditional discovery-based site quality values that convey confidence that the site is anything besides the homozygous reference genotype, such as SNV or quality, are not used. Instead, a site, or locus, is filtered based on the confidence in the reported genotype for the current sample.

The genotype quality used in gVCF is a Phred-scaled probability that the given genotype is correct. It is indicated with the FORMAT field tag GQX. Any locus where the genotype quality is below the cutoff threshold is filtered with the tag LowGQX. Besides *filtering on genotype quality*, other filters can also be applied.

The gVCF output from Strelka Germline Variant Caller includes several heuristic filters applied to the site and indel records. The filters are as follows.

Table 2 VCF Site and Indel Record Filters

VCF Filter ID	Type	Description
PhasingConflict	site	The locus read evidence displays unbalanced phasing patterns.
Ploidy Conflict	site/indel	The genotype call from the variant caller is not consistent with the chromosome ploidy.
IndelConflict	indel	The locus is in region with conflicting indel calls.
HighDPFRatio	site	The fraction of basecalls filtered out at a site, DPF/ (DP+DPF), is greater than 0.4.
HighDepth	site/indel	The locus depth is greater than 3x the mean chromosome depth.
LowGQX	site/indel	Locus GQX is less than 30.
LowGQXHetSNP	site	Locus GQX is less than 15 for het SNP.
LowGQXHomSNP	site	Locus GQX is less than 17 for hom SNP.
LowGQXHetIns	site/indel	Locus GQX is less than 6 for het insertion.
LowGQXHomIns	site/indel	Locus GQX is less than 6 for hom insertion.

VCF Filter ID	Type	Description
LowGQXHetDel	site/indel	Locus GQX is less than 6 for het deletion.
Pass	site/indel	Position has passed all filters.
SiteCOnflict	indel	The site genotype conflicts with the proximal indel call. This call is typically a heterozygous SNV call made inside of a heterozygous deletion.

Canvas

Canvas is an algorithm that calls copy number variants from a diploid sample or a matched pair of tumor and normal samples. Most of a normal DNA sample is diploid, which means having paired chromosomes. Canvas identifies regions of the sample genome that are either not present or are present 1 time or more than 2 times in the genome.

Canvas scans the genome for regions with an unexpected number of short read alignments. Regions with fewer than the expected number of alignments are classified as losses. Regions with more than the expected number of alignments are classified as gains.

Canvas is intended for low-depth cytogenetics experiments, low-depth single-cell experiments, or whole-genome sequencing experiments. Canvas is not appropriate for an experiment with the following conditions:

- Most of the genome is not assumed to be diploid.
- Reads are not distributed randomly across the diploid genome.

For more information, visit github.com/Illumina.canvas.

Workflow

Canvas can be conceptually divided into 4 processes:

- Binning—Counting alignments in genomic bins.
- Cleaning—Removal of systematic biases and outliers from the counts.
- Partitioning—Partitioning the counts into homogenous regions.
- Calling—Assigning a copy number to each homogenous region.

These processes are explained in subsequent sections.

Binning

The binning procedure creates genomic windows, or bins, across the genome and counts the number of observed alignments that fall into each bin. The alignments are provided in the form of a BAM file.

Canvas binning keeps in memory a collection of BitArrays to store observed alignments, one BitArray for each chromosome. Each BitArray length is the same as its corresponding chromosome length. As the BAM file is read in, Canvas records the position of the left-most base in each alignment within the chromosome-appropriate BitArray. After all alignments in the BAM file have been read, the BitArrays have a 1 wherever an alignment was observed and a 0 everywhere else.

After reading in the BAM file, a masked FASTA file is read in, one chromosome at a time. This FASTA file contains the genomic sequences that were used for alignment. Each 35-mer within this FASTA file is marked as unique or nonunique with uppercase and lowercase letters. If a 35-mer is unique, then its first nucleotide is capitalized; otherwise, it is not capitalized. For example, in the sequence:

```
acgtttaATgacgatGaacgatcagctaagaatacgacaatatcagacaa
```

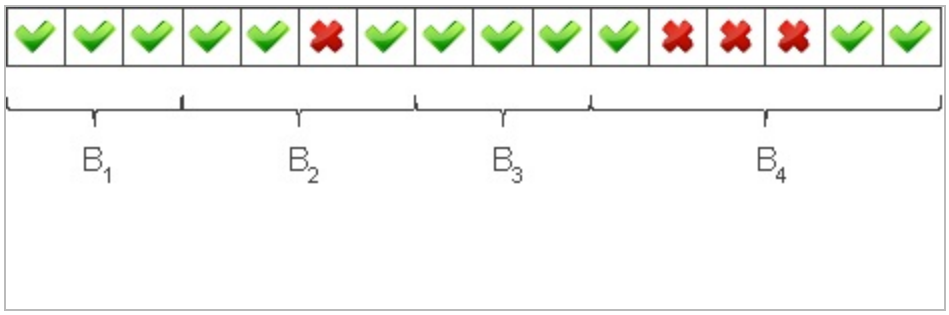
The 35-mers marked as unique are as follows:

```
ATGACGATGAACGATCAGCTAAGAATACGACAATA
TGACGATGAACGATCAGCTAAGAATACGACAATAT
GAACGATCAGCTAAGAATACGACAATATCAGACAA
```

Canvas stores the genomic locations of unique 35-mers in another collection of BitArrays analogous to BitArrays used to store alignment positions. Unique positions and nonunique positions are marked with 1s and 0s, respectively. This marking is used as a mask to guarantee that only alignments that start at unique 35-mer positions in the genome are used.

Bin Sizes

Canvas is initialized with 100 alignments per bin and then proceeds to compute the bin boundaries such that each bin contains the same bin size, or number of unique 35-mers. The term "bin size" refers to the number of unique genomic 35-mers per bin. Because some regions of the human genome are more repetitive than others, physical bin sizes (in genomic coordinates) are not identical. In the following example, each box is a position along the genome. Each checkmark represents a unique 35-mer while each X represents a nonunique 35-mer. The bin size in this example is 3 (3 checkmarks per bin). The physical size of each bin is not constant. B1 and B3 have a physical size of 3 but B2 and B4 have physical sizes of 4 and 6, respectively.



Computing Bin Size

To compute bin size, the ratio of observed alignments to unique 35-mers is calculated for each autosome. The desired number of alignments per bin is then divided by the median of these ratios to yield bin size. For whole-genome sequencing, bin sizes are typically in the range of 800–1000 unique 35-mers. Correspondingly, most physical window sizes are in the 1–1.2 kb range. The advantage of this approach relative to using fixed genomic intervals is that the same number of reads map to each bin, regardless of "uniqueness" or ability to be mapped.

After bin size is computed, bins are defined as consecutive genomic windows such that each bin contains the same bin size, or number of unique 35-mers. The number of observed alignments present within the boundary of each bin is then counted from the alignment BitArrays. The GC content of each bin is also calculated. The chromosome, genomic start, genomic stop, observed counts and GC content in each bin are output to disk.

Cleaning

Canvas cleaning comprises the following 3 procedures that remove outliers and systematic biases from the count data computed in the caller.

1 Single point outlier removal.

2 Physical size outlier removal.

3 GC content correction.

These procedures are performed on the bins produced during the Canvas binning process.

Single Point Outlier Removal

This step removes individual bins that represent extreme outliers. These bins have counts that are very different from the counts present in upstream and downstream bins. Two values, a and b, are defined as to be very different when their difference is greater than expected by chance, assuming a and b come from the same underlying distribution. These values use the Chi-squared distribution, as follows:

- $\mu = 0.5a + 0.5b$
- $\chi 2 = ((a - \mu)^2 + (b - \mu)^2)\, \mu\text{-}1$

A value of $\chi 2$ greater than 6.635, which is the 99th percentile of the Chi-squared distribution with 1 degree of freedom, is considered very different. If a bin count is very different from the count of both upstream and downstream neighbors, then the bin is deemed an outlier and removed.

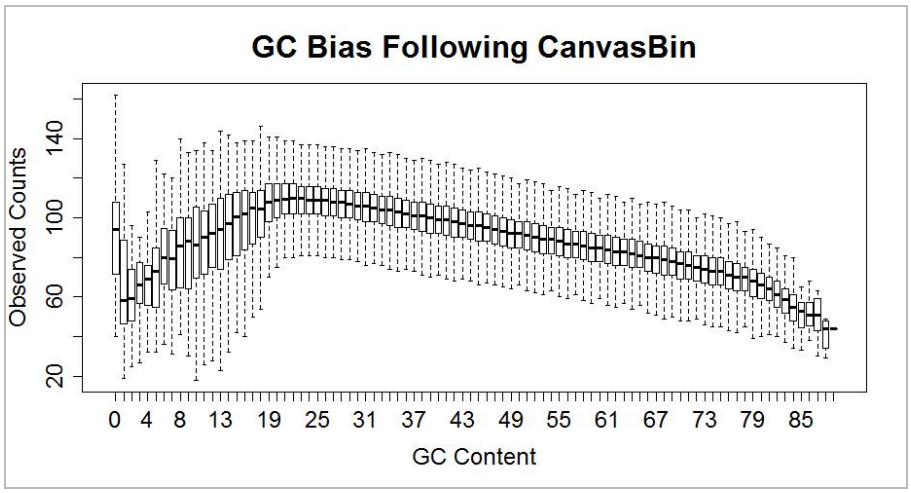
Physical Size Outlier Removal

Bins likely do not have the same physical (genomic) size. The average for whole-genome sequencing runs might be approximately 1 kb. If the bins cover repetitive regions of the genome, some bins sizes might be several megabases in size. Example regions might include centromeres and telomeres. The counts in these regions tend to be unreliable so bins with extreme physical size are removed. Specifically, the 98th percentile of observed physical sizes is calculated and bins with sizes larger than this threshold are removed.

GC Content Correction

The main variability in bins counts is GC content. An example of the bias is represented in the following figure.

Figure 3 GC Bias Example



The following correction is performed:

1 Bins are first aggregated according to GC content, which is rounded to the nearest integer.

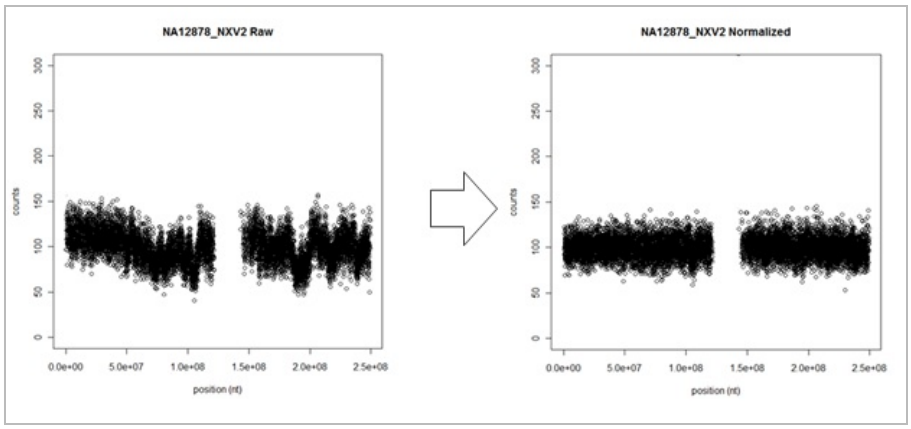
2 Second, each bin count is divided by the median count of bins having the same GC content.

3 Finally, this value is multiplied by the desired average count per bin (100 by default) and rounded to the nearest integer. The effect is to flatten the midpoints of the bars in the example box-and-whisker plot.

Some values for GC content have few bins so the estimate of its median is not robust. Therefore, bins are discarded when the number of bins having the same GC content is fewer than 100.

For some sample preparation schemes, GC content correction has a dramatic effect. The following figure illustrates the effect of GC content correction for a low depth sequencing experiment using the Nextera library preparation method. The figure on the left shows bins counts as a function of chromosome position before normalization. The figure on the right shows the result after GC content correction.

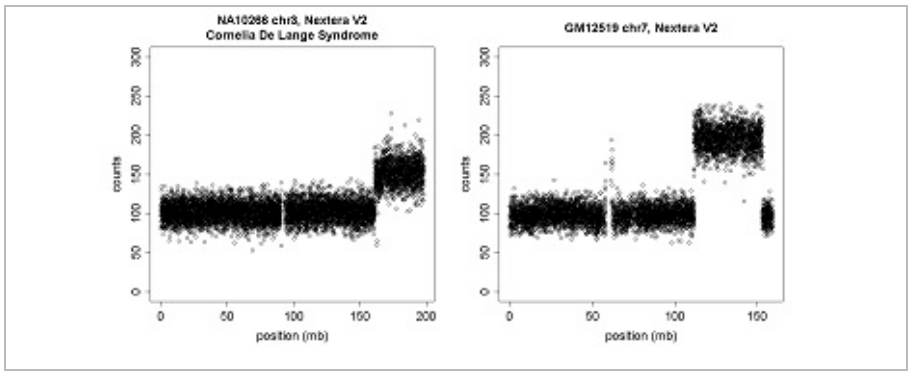
Figure 4 GC Content Correction



For whole-genome sequencing experiments, the typically median absolute deviations (MADs) are 10.3, which is close to the expected value of 10. The expected value is predicted using the Poisson model for an average count of 100 and indicates that little bias remains following GC content correction.

It is important to note that the normalization signal does not dampen signal from CNVs as shown in the following 2 figures. The figure on the left shows a chromosome known to harbor a single copy gain. The figure on the right shows chromosome known to harbor a double copy gain.

Figure 5 Chromosomal Copy Gain



Partitioning

Canvas partitioning implements an algorithm for identifying regions of the genome such that their average counts are statistically different than average counts of neighboring regions. The implementation is a port of the circular binary segmentation (CBS) algorithm.

The algorithm is fully described in (Olshen AB, 2004) and (Venkatraman ES, 2007). The algorithm briefly considers each chromosome as a segment. The algorithm assesses each segment and identifies the pair of bins for which the counts in the bins between them are maximally different than the counts of the rest of the bins. The statistical significance of

the maximal difference is assessed via permutation testing. If the difference is statistically significant, then the procedure is applied recursively to the 2 or 3 segments created by partitioning the current segment by the identified pair of points. Input to the algorithm is the output generated by the Canvas cleaning algorithm.

Because of the computational complexity of the algorithm $O(N^2)$, the problem is divided into subchromosome problems followed by merging, in practice. Heuristics are used to speed up the permutation testing.

Calling

The final module of the Canvas algorithm is to assign discrete copy numbers to each of the regions identified by the Canvas partitioner.

A Gaussian model is used as the default calling method. In this case, both the mean and standard deviation are estimated from the data for the diploid model and adjusted for the other copy number models. For example, if the mean, μ, and standard deviation, σ, are estimated to be 100 and 15 in the diploid model, then corresponding estimates in the haploid model would be μ/2 and σ/2. The mean and standard deviation are estimated using the autosomal median and MAD of counts. This model is the default as it is more appropriate in cases where the spread of counts is higher than expected from the Poisson model due to unaccounted sources of variability. An example of this case is single cell sequencing experiments where whole-genome amplification is required.

Following assignment of copy number states, neighboring regions that received the same copy number call are merged into a single region.

Phred-scaled Q-scores are assigned to each region using a simple logistic function derived using array CGH data as the gold standard. The probability of a miscall is modeled as

p=1-(1/((1+e^(0.5532-0.147N)))

Where N is the number of bins found within the nondiploid region. This probability is converted to a Q-score by

q=-10 log p

This estimate is likely conservative as it is derived from array CGH. Importantly, Q-scores are a function of number of bins, not genomic size, so they are applicable to experiments of any sequencing depth, including low-depth cytogenetics screening.

The coordinates of nondiploid regions and their Q-scores are output to a VCF file. Two filters are applied to PASS variants. First, a variant must have a Q-score of Q10 or greater. Second, a variant must be of size 10 kb, or greater.

Manta (Structural Variant Caller)

Manta is a structural variant caller for short sequencing reads. It can discover structural variants of any size and score these variants using both a diploid genotype model and a somatic model (when separate tumor and normal samples are specified). Structural variant discovery and scoring incorporate both paired read fragment spanning and split read evidence.

Method Overview

Manta works by dividing the structural variant discovery process into 2 primary steps–scanning the genome to find SV associated regions and analysis, scoring, and output of SVs found in such regions.

1 **Build SV association graph**
Scan the entire genome to discover evidence of possible SVs and large indels. This evidence is enumerated into a graph with edges connecting all regions of the genome that have a possible SV association. Edges can connect 2 different regions of the genome to represent evidence of a long-range association, or an edge can connect a region to itself to capture a local indel/small SV association. These associations are more general than a specific SV hypothesis, in that many SV candidates can be found on 1 edge, although typically only 1 or 2 candidates are found per edge.

2 **Analyze graph edges to find SVs**
Analyze individual graph edges or groups of highly connected edges to discover and score SVs associated with the edges. The substeps of this process include:
- Inference of SV candidates associated with the edge.
- Attempted assembly of the SVs break-ends.
- Scoring and filtration of the SV under various biological models (currently diploid germline and somatic).
- Output to VCF.

Capabilities

Manta can detect all structural variant types that are identifiable in the absence of copy number analysis and large scale *de novo* assembly. Detectable types are enumerated in this section.

For each structural variant and indel, Manta attempts to align the break-ends to base pair resolution and report the left-shifted break-end coordinate (per the VCF 4.1 SV reporting guidelines). Manta also reports any break-end microhomology sequence and inserted sequence between the break-ends. Often the assembly fails to provide a confident explanation of the data. In such cases, the variant is reported as IMPRECISE, and scored according to the paired-end read evidence alone.

The sequencing reads provided as input to Manta are expected to be from a paired-end sequencing assay that results in an inwards orientation between the 2 reads of each DNA fragment. Each read presents a read from the outer edge of the fragment insert inward.

Detected Variant Classes

Manta is able to detect all variation classes that can be explained as novel DNA adjacencies in the genome. Simple insertion/deletion events can be detected down to a configurable minimum size cutoff (defaulting to 51). All DNA adjacencies are classified into the following categories based on the break-end pattern:

- Deletions
- Insertions
- Inversions
- Tandem Duplications
- Interchromosomal Translocations

Known Limitations

Manta cannot detect the following variant types:

- Nontandem repeats/amplifications
- Large insertions—The maximum detectable size corresponds to approximately the read-pair fragment size, but note that detection power falls off to impractical levels well before this size.
- Small inversions—The limiting size is not tested, but in theory detection falls off below ~200 bases. So-called microinversions might be detected indirectly as combined insertion/deletion variants.

More general repeat-based limitations exist for all variant types:

- Power to assemble variants to break-end resolution falls to 0 as break-end repeat length approaches the read size.
- Power to detect any break-end falls to (nearly) 0 as the break-end repeat length approaches the fragment size.
- The method cannot detect nontandem repeats.

While Manta classifies novel DNA-adjacencies, it does not infer the higher level constructs implied by the classification. For instance, a variant marked as a deletion by Manta indicates an intrachromosomal translocation with a deletion-like break-end pattern. However, there is no test of depth, b-allele frequency, or intersecting adjacencies to infer the SV type directly.

Methods for Tumor-Normal Workflow

This section describes the underlining methodologies for the Tumor-Normal analysis algorithm in HiSeq Analysis Software v2.1.

Overview

The somatic variant calling pipeline uses a normal BAM file and a tumor BAM file as input.

These BAM files are then processed through 3 interconnected callers:

- Strelka (Somatic Variant Caller)
- Manta (Structural Variant Caller)
- Canvas (Somatic Copy Number Variant Caller)

During the first stage of the pipeline, the tumor and normal BAM files run through a combined indel realignment operation. This realignment operation is used as the input for further processing. During calling, putative calls and *de novo* reassembled sections of sequence are passed between the callers to produce internally consistent variant calls.

All 3 callers use statistical models that operate on the combined tumor and normal reads as input instead of the variants. The statistical models use combined calling instead of subtraction of variant calls. Using combined calling produces superior results. However, subtraction of the calls from the normal and tumor whole genome results often do not match the somatic calls from a combined caller. For example, you can find a somatic variant that was not called in the tumor WGS sample because the combined caller is operating on the reads.

Strelka (Somatic Variant Caller)

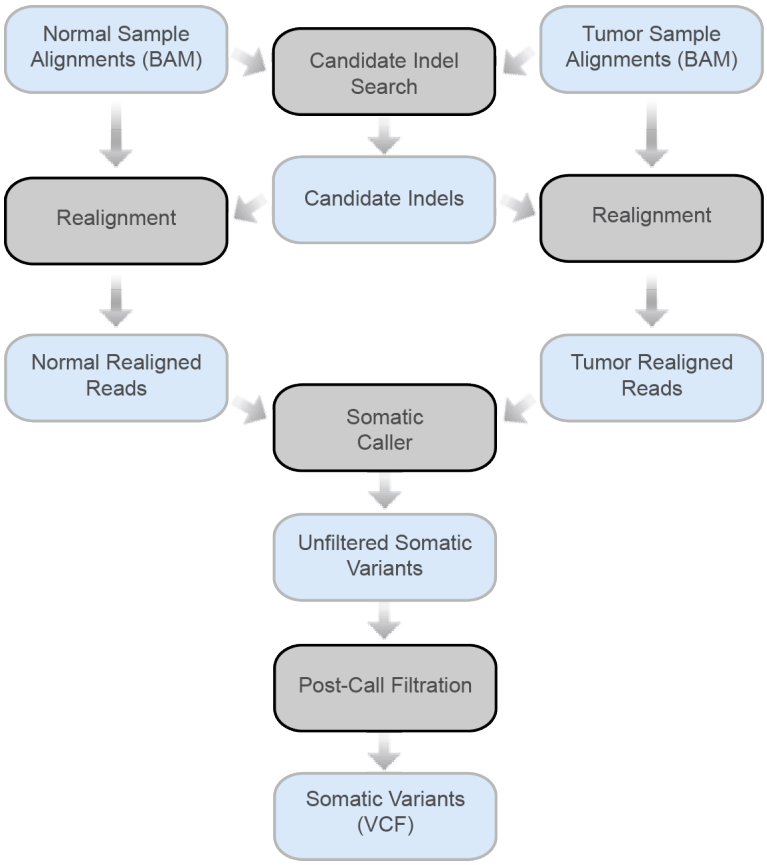
Strelka detects somatic SNVs and indels in sequencing data from a tumor and matched normal sample.

The analysis is based on the following assumptions:

- The normal sample is a mixture of diploid germline variation and noise.
- The tumor sample is a combination of the normal sample and somatic variation. It is assumed that the somatic variation and the normal noise can occur at any allele frequency ratio.

For SNVs, but not for indels, the normal noise component is further modeled as a combination of single-strand and double-strand noise.

Figure 6 Strelka Method



NOTE

For a detailed overview of Strelka methods, go to www.ncbi.nlm.nih.gov/pubmed/22581179.

Saunders,C.T., Wong,W.S., Swamy,S. *et al.* (2012) Strelka: accurate somatic small-variant calling from sequenced tumor-normal sample pairs. *Bioinformatics*, 28, 1811–1817. doi: 10.1093/bioinformatics/bts271

Candidate Indel Search

Strelka Somatic Variant Caller scans through the genome using sequence alignments from the normal sample and tumor sample together to find a joint set of candidate indels. The information in sequence alignments is supplemented with externally generated candidate indels discovered by Canvas. Canvas provides external candidate indels to Strelka for indels of size 50 and below.

Candidate indels are used for realignment of reads, during which each candidate indel is evaluated as a potential somatic indel. Any other types of indels are considered noise indels. If a better alignment is not found, these indels are allowed to remain in the read alignments; otherwise, they are not used.

The candidate indel thresholds are designed so that the joint candidate indel set is at least the combined set found if Strelka Germline Variant Caller is run on the individual samples. Specifically, where a minimum number of nominating reads is required for candidacy in Strelka Germline Variant Caller, Strelka Somatic Variant Caller requires the same minimum number of nominating reads from the combined input. Strelka Somatic Variant Caller requires that at least 1 sample contains a minimum fraction of supporting reads among the sample reads for candidacy.

Realignment

For every read that intersects a candidate alignment, the Strelka Somatic Variant Caller attempts to find the most probable alignments including the candidate indel and excluding the candidate indel. Typically, the alignment excluding the candidate indel aligns to the reference, but occasionally an alternate indel that overlaps or interferes with the candidate is found to be more likely. The indel caller uses the probabilities of both alignments as part of the indel quality score calculation, whereas only a single alignment (usually the most probable) is preserved for SNV calling.

Somatic Caller

Strelka Somatic Variant Caller uses a Bayesian probability model similar to the one used for germline variant calling in Strelka Germline Variant Caller, or in external tools such as GATK. Using this model, our objective is to compute the posterior probability $P(\theta \mid D)$, which is the probability of the model state θ conditioned on the observed sequencing data.

In a germline variant caller, the state space of the model is conventionally a discrete set of diploid genotypes. For SNVs, the set of possible states is:

G={"AA,CC,GG,TT,AC,AG,AT,CG,CT,GT"}

The Strelka Somatic Variant Caller model instead approximates continuous allele frequencies for each allele:

f={f_A, f_C, f_G, f_T}

The allele frequencies are restricted to allow a maximum of 2 nonzero frequencies. Any additional alleles observed in the data are treated as noise.

Another departure from typical germline calling methods is that the state space of the model is the allele frequency of both the tumor and the normal sample. In the following equation, f_t and f_n represent the allele frequencies of the tumor and normal samples, respectively.

θ=(f_t, f_n)

The final somatic variant quality value reported by the model is computed from the probability that the allele frequencies are unequal (ie, f_t≠f_n) given the observed sequence data.

Post-Call Filtration

Heuristic filters remove several types of improbable calls resulting from data artifacts that cannot be easily represented in the somatic probability model. These filters act as a final step to separate out the somatic calls reported by Strelka.

Input Data Filtration

Strelka uses 2 tiers of input data filtration during somatic small variant calling:

- Tier 1—A more stringent filtering to ensure high quality calls
- Tier 2—A lower filtration stringency

Initially, candidates are called using a subset of the data with more stringent tier 1 filtering. If the method produces a nonzero quality score for any SNV or indel, the potential somatic variant is called again using data with a lower tier 2 stringency. The lower quality from the 2 tiers is selected for output. However, if the tier 2 quality is 0, the call is eliminated.

For somatic SNVs and indels, Strelka produces a general somatic quality score, Q (ssnv), or Q (somatic indel). This score indicates the probability of the somatic variant and a joint probability of the somatic variant and a specific normal genotype, Q (ssnv+ntype), or Q (somatic indel + ntype). The 2 tier evaluation is applied to each of these qualities separately, as follows:

- Q (ssnv) = min (Q (ssnv|tier1), Q (ssnv|tier2))

- Q (ssnv+ntype) = min (Q (ssnv+ntype|tier1), Q (ssnv+ntype|tier2))

The tier used for each quality value is provided in the Strelka output record for each somatic variant. If the most likely normal genotype is not the same at tier 1 and tier 2, then the normal genotype is reported as a conflict in the output.

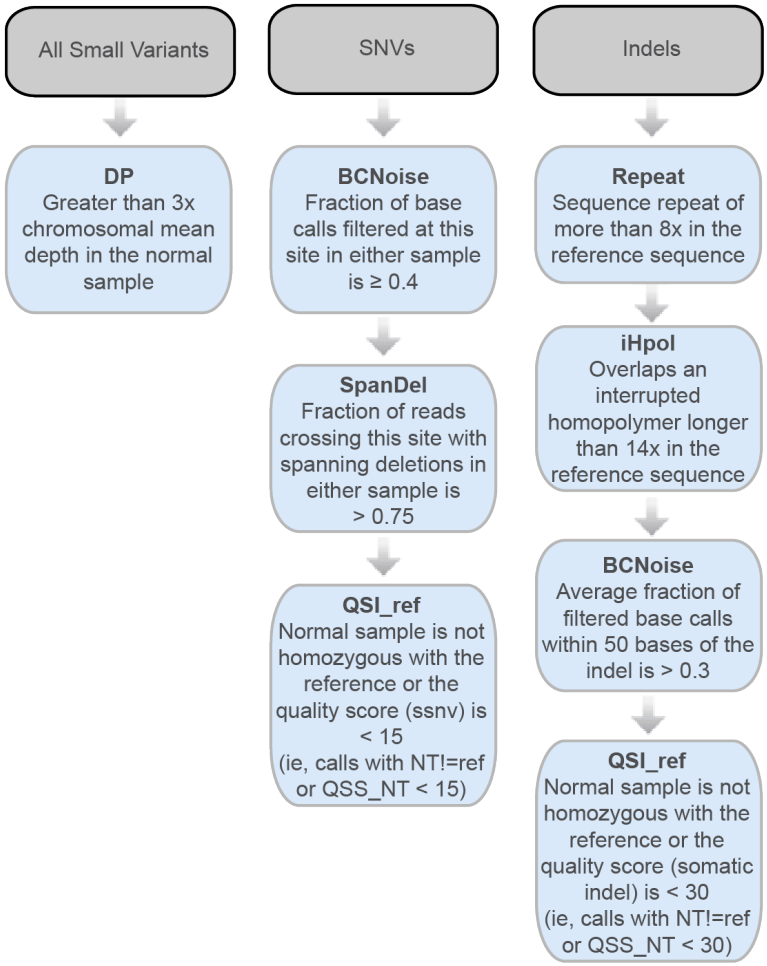
Using 2 data tiers enables an initial somatic call based on high-quality data. Given a potential call, using 2 data tiers removes support for the putative somatic allele in the normal sample from lower quality data. The following table lists the primary data filtration levels that are changed between tier 1 and tier 2.

Parameter	Tier 1 Value	Tier 2 Value
Min paired-end alignment score	20	0
Min single-end alignment score	10	0
Single-end score rescue?	No	Yes
Include unanchored pairs?	No	Yes
Include anomalous pairs?	No	Yes
Include singleton pairs?	No	Yes
Mismatch density filter—Maximum mismatches in window	3	10

Additional Filtration

After the somatic filter is finished, more filters are applied. A single candidate somatic call can be annotated with several filters.

Figure 7 Additional Filtration



Quality Filtration Levels

Only somatic calls originating from homozygous reference alleles in the normal sample are reviewed for validation and included in the output.

- Somatic SNVs are reported if the normal genotype is equal to the reference *and* Q (ssnv+ntype) ≥ 15.
- Somatic indels are reported if the normal genotype is equal to the reference *and* Q (somatic indel+ntype) ≥ 30.

NOTE

The value Q (ssnv+ntype) is associated with the VCF key QSS_NT.

The value Q (somatic indel+ntype) is associated with the VCF key QSI_NT.

CanvasSomaticCaller (Somatic Copy Number Variations Caller)

Canvas is a tool for calling copy number variants (CNVs) from human DNA sequencing data. The CanvasSomaticCaller mode works with paired tumor/normal samples. The primary input is aligned reads in BAM format. The primary output is a report VCF file that gives the copy number status of the genome.

For more information about Canvas, see *Canvas* on page 29 or the Canvas GitHub page.

Revision History

Part #	Date	Description of Change
Document # 15070536 v01	January 2017	Updated the software name to HiSeq Analysis Software v2.1. Updated the FASTQ folder structure.
Document # 15070536 Rev. A	May 2015	Initial Release.

Technical Assistance

For technical assistance, contact Illumina Technical Support.

Website: www.illumina.com
Email: techsupport@illumina.com

Illumina Customer Support Telephone Numbers

North America 1.800.809.4566	**Germany** 0800.180.8994	**Singapore** 1.800.579.2745
Australia 1.800.775.688	**Hong Kong** 800960230	**Spain** 900.812168
Austria 0800.296575	**Ireland** 1.800.812949	**Sweden** 020790181
Belgium 0800.81102	**Italy** 800.874909	**Switzerland** 0800.563118
China 400.635.9898	**Japan** 0800.111.5011	**Taiwan** 00806651752
Denmark 80882346	**Netherlands** 0800.0223859	**United Kingdom** 0800.917.0041
Finland 0800.918363	**New Zealand** 0800.451.650	**Other countries** +44.1799.534000
France 0800.911850	**Norway** 800.16836	

Safety data sheets (SDSs)—Available on the Illumina website at support.illumina.com/sds.html.